Brock University

Department of Computer Science

Automatic Inference of Hierarchical Graph Models Using Genetic Programming with an Application to Cortical Networks

Alexander Bailey, Beatrice Ombuki-Berman, and Mario Ventresca
Technical Report # CS-13-03
March 2013

Brock University
Department of Computer Science
St. Catharines, Ontario
Canada L2S 3A1
www.cosc.brocku.ca

Automatic Inference of Hierarchical Graph Models using Genetic Programming with an Application to Cortical Networks

Alexander Bailey
Brock University
St. Catharines, Canada
ab04bf@brocku.ca

Beatrice Ombuki-Berman
Brock University
St. Catharines, Canada
bombuki@brocku.ca

Mario Ventresca
University of Toronto
Toronto, Canada
mario.ventresca@utoronto.ca

ABSTRACT

The pathways that relay sensory information within the brain form a network of connections, the precise organization of which is unknown. Communities of neurons can be discerned within this tangled structure, with inhomogeneously distributed connections existing between cortical areas. Classification and modelling of these networks has led to advancements in the identification of unhealthy or injured brains, however, the current models used are known to have major deficiencies. Specifically, the community structure of the cortex is not accounted for in existing algorithms, and it is unclear how to properly design a more representative graph model. It has recently been demonstrated that genetic programming may be useful for inferring accurate graph models, although no study to date has investigated the ability to replicate community structure. In this paper we propose the first GP system for the automatic inference of algorithms capable of generating, to a high accuracy, networks with community structure. We utilize a common cat cortex data set to highlight the efficacy of our approach. Our experiments clearly show that the inferred graph model generates a more representative network than those currently used in scientific literature.

1. INTRODUCTION

The term *complex network* [23] refers to any set of interrelated elements where the pattern of connections is meaningful. The elements are the vertices in the network and their connections form the edges. Usually we are interested in network behaviour emergent from these patterns and by studying how these networks connect it is possible to learn the functional implication of these connections. In this paper we will focus our attention on the cortical network of a cat. The study of the brain as a complex network is relatively new, and began with the advent of modern technologies that facilitate the large-scale mapping of brain connectivity [30].

Numerous studies have focused on various aspects of brain connectivity and their implications on its functionality and health. These studies range from very large scale simulations of neuron growth and interaction requiring petascale computing power [19], to examining the connectivity of high-level regions of the brain associated with functional characteristics – in particular the cortex [20, 30, 31, 33, 36]. Existing human-designed graph models have provided valuable insights into the organization of cortical networks, and have even helped to refine methods of diagnosis for neurological diseases [5, 18]. However, existing commonly-used graph models do not describe the connectivity between functionally separate cortical areas, and there is some debate about which of these models is best to use given that none of them properly describe the network dynamics [36]. Simulations of the brain at the micro-level could potentially provide very accurate models of its large-scale organization, however, they are inaccessible to most researchers due to the computational resources required. Simulations of a rat cortical column consisting of about 10^3 neurons have successfully been constructed using modern supercomputing technology [19] – the scale of a full cortical simulation, the human cortex has an estimated 10^{10} neurons, making such a simulation currently impossible. Simple, accessible, and accurate models or methods of producing models of the high level organization of cortical networks are needed to further our understanding of their function.

Recently, it has been shown that Genetic Programming (GP) is a promising candidate for the automatic inference of graph models for complex networks [3]. We propose a model of hierarchical cortical connectivity inferred automatically via GP, and show that it performs at least as well or better than existing popular models of cortical structure.

2. BACKGROUND

Complex networks are usually studied in terms of their overall structure using statistical measures of connectivity. When considering cortical networks the *average geodesic path length*, as well as the *clustering coefficient* sometimes called *transitivity*, the *degree distribution*, and the *community structure* are of interest because of their effect on communication pathways ([1,16,24] provide a good overview of these terms). Various studies focusing on categorizing cortical networks include [8, 29, 30, 32]. Throughout this paper the assumed definition of a graph is $G = (V, E)$ where V is the vertex set and E is the edge set, the size of the vertex set $|V| = n$, and the degree of vertex i is k_i. The transitivity and the average geodesic path length will be referred to as C and l respectively.

2.1 Complex Network Models

Community structure [16] is a property found in many networks, in which there are subsets of nodes that are more connected to each other than the rest of the network. The densely connected subsets are known as communities. Community structure is apparent in cortical networks, is known to correspond to functional groups of cortical nodes, and injured and unhealthy brains have been shown to exhibit

different community structures than healthy ones [5, 8, 11, 26, 32, 36].

A value related to the community structure is the centrality of a vertex, which describes how important or connected they are within a network [24]. The *betweenness centrality* of a vertex is defined as the number of geodesic paths that run through it. The betweenness centrality describes a vertex's influence on the flow of information within a network and is expected to be an important property with respect to cortical networks [32].

Given information about a network with respect to the properties described above, algorithms can be designed that produce a potentially infinite set of graphs that mimic these properties, these algorithms are called *graph models*. Cortical networks, having a low average geodesic path length and a high clustering coefficient are often modelled with the Watts-Strogatz (WS) small-world model [26, 31, 32, 35]. However, the WS model does not produce highly connected hub nodes, an important feature of brain structure. The hub nodes present in cortical networks have given rise to the idea that the connectivity may follow a power law. Networks that possess this property are called *scale-free* (SF) and if it is necessary to model this property then a model such as the Barabási-Albert (BA) is usually used [4, 26, 32]. Both of these models have been important in the analysis of brain networks [32], however neither of them model both the degree distribution as well as the transitivity and path length properties.

2.2 Cortical Networks

The experiments in this paper will focus on the cortical connectivity within the brain of a cat because it has the most complete available data of its kind [36]. The dataset was created via a collation of tract-tracing experiments and literature reporting, supported by analytical treatment of the results [28]. It has been found that this network is dominated by paths of length one and two, that it contains a high density of connections, it has highly connected hubs, and that it is segregated into clusters (which closely align with the visual, auditory, somatosensory-motor, and frontolimbic centres) [36]. It is also speculated that the number of alternative pathways between cortical nodes plays an important role in the brain's ability to process complex information [36]. Cortical networks are often classified as small-world networks [31, 36]; however, there is also some debate as to whether or not they could be scale-free networks [17, 20, 33]. As such, these networks are often modelled or simulated using algorithms that generate scale-free or small-world networks. Although, it is known that the cortical data in the literature suggests neither model is quite suitable given that they do not describe the inter-community dynamics of the cortex well [36].

The proposed system for the automatic inference of graph models works only with undirected networks, and while cortical networks are thought to be directed networks, it has been shown that an undirected approximation is a reasonable relaxation because the majority of connections are reciprocal [22, 36]. The cat cortical network dataset [28] in its undirected form contains 52 cortical nodes and 515 edges. It has a clustering coefficient of $C_c = 0.585$ and an average geodesic path length of $l_c = 1.636$ and a diameter of $d_c = 3$. These simple properties are useful because they are known to be good classifiers of the behaviour of message propagation across a network [2]. If multiple paths are used for processing then it may be that the longest path is also of consequence, especially if the temporal aspect of these signals is important, so the diameter will also be considered.

3. GP SYSTEM

The GP system for automatically generating graph models for complex networks was built using *RobGP*, a Koza-style GP system [13]. All operations were strongly typed. The system generates hierarchical graph models given an initial set of input networks, called the *target networks*. The set can be any size, but they should all be related as only one algorithm will be generated that should approximate all targets in the set. Hierarchical models generated by the GP are meant to exhibit a strong community structure. This is achieved by breaking the target apart into two sets of subgraphs, which are used as target sets for two parallel evolving populations. These populations are distinct but use the same GP language, fitness functions, parameters, and are evaluated in the same way. This process is described in the following subsection.

The goal of the application of the GP system is to automatically infer a model of the cortex data which exhibits an accurate degree distribution, average geodesic path length, transitivity, and exhibits a community structure which mimics the structure of the cortex. Neither the BA nor WS model exhibit all of these features, and although it is a desirable property, none of them generate graphs with a community structure that matches the cortex [26].

3.1 Generation of Hierarchical Models

This paper does not focus on the problem of discovering communities directly with the GP system itself, although it may be an effort for future work. Instead, we endeavour to model communities that we are able to identify in a target network via an external algorithm (many exist [12]). We propose a system which represents the first effort to automatically infer graph models for networks which exhibit a hierarchical clustered organization for a defined community structure. It works as follows:

1. A black-box method is employed to identify the communities within the target network.
2. The communities are isolated and fed into the GP as the target set for the community model population.
3. The inter-community edges form a graph which is fed into the GP as the target for the inter-community model population.
4. Select two models, one evolved from each population, and combine them to create a hierarchical model.

Note that Steps 2 and 3 are performed in parallel. Step 4 is accomplished by using the community model to generate a number of small graphs $\{C_1, .., C_l\}$ that are combined into a graph G_C using a disjoint union. Next a graph, G_O, is generated by the inter-community model using an initially empty edge set, and the vertex set of G_C. The final graph is then constructed $G = \{V(G_C), E(G_C) \cup E(G_O)\}$ where $V(G_C)$, $E(G_C)$, and $E(G_O)$ are the vertex set for G_C, and the edge sets for G_C and G_O respectively. The process of selecting a model from each population for Step 4 will be

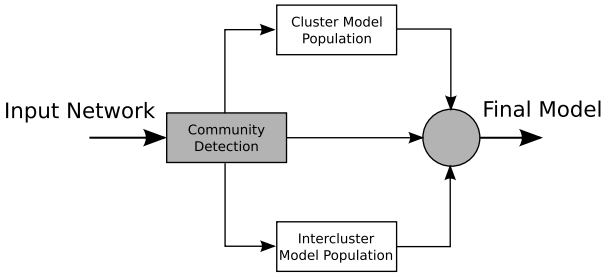


Figure 1: The process of generating hierarchical models.

described in Section 3.3, but some necessary definitions must first be made in Section 3.2. Figure 1 illustrates the steps above.

The community detection algorithm is responsible only for dividing the initial target graph and informing the final model about how many communities it should create. The evolved model is responsible for deciding how it initializes a graph, in what way it adds nodes, how to connect those nodes, and how many edges to do it with. It should also be noted that the community detection algorithm could be exchanged for any method which divides the original network. In this paper, the algorithm used is the leading eigenvector community detection algorithm [25].

There are some issues for further investigation in the proposed method, in particular the number of communities is dependent on a the community detection algorithm external to the GP system. Secondly, the GP system is aware of how the community and inter-community models perform independently, but their interaction is not evaluated. Hence, while a future study will evaluate what the best method for hierarchical model generation is, and further address these issues, this paper represents the first effort at automatically generating these types of models and our preliminary results already outperform existing models.

3.2 Selection and Fitness Evaluation

The fitness functions were designed to compare a target network to the graphs produced by the evolved models, that we call the *active graphs*. Graph models are usually benchmarked by comparing how well they reproduce a particular *feature* of a set of complex networks [4, 23, 35]. There is evidence to suggest that the degree distribution, the average geodesic path length, and the clustering coefficient are sufficient to describe nearly all of the diffusive behaviour on a network [2], thus our fitness functions are based on comparing these features between the active and target graph. Our system uses a multi-objective, weighted and normalized, *summed-ranks* strategy [6] to consider multiple features. This multi-objective strategy is simple to use, ranks are easy to compute, it has been shown to perform well in problems of high dimensionality [10], and recently it has been shown to work well on problems of low dimensionality [7, 14, 27].

Summed-ranks works by assigning each individual j a raw fitness value for each objective i. Individuals are then assigned a rank $Rank_{ji}$ with respect to each objective, such that zero is the worst rank. Assuming a maximization problem, the fitness F_j of an individual is defined as:

$$F_j = \frac{\sum_{i \in O} w_i \times Rank_{ji}}{\max(Rank_i)} \tag{1}$$

where O is the set of objectives and w_i is the weight placed at objective i. This is the same formulation given in [6] for their *Weighted Average Ranking* with the addition of a rank normalization by the maximum rank in each objective. The raw fitness objectives are defined as:

$$F_1Raw = |l(G_t) - l(G_a)| \tag{2}$$

$$F_2Raw = [C(G_t) - C(G_a)]^2 \tag{3}$$

$$F_3Raw = D_{G_t,G_a} \tag{4}$$

$$F_4Raw = \frac{1}{|H|} \sum_i \frac{(h_i^t - h_i^a)}{h_i^t + h_i^a}. \tag{5}$$

Where $l(G_t)$ is the average geodesic path length of the target graph, $l(G_a)$ is the average geodesic path length of an active graph, $C(G_t)$ and $C(G_a)$ are global clustering coefficients of the target and an active graph respectively, and $D_{t,a}$ is the K-S test [9] statistic comparing the degree distribution of the target graph to an active graph. $|H|$ is the number of discrete values in the degree distribution histogram, h_i^t and h_i^a are the i^{th} values in the target network and active graph's degree distributions respectively.

The raw fitnesses are used during evolution in order to compute the ranks of each model. For presentation, they are adjusted to values in the range $[0, 1]$ with a value of 1 being the most desirable. These adjusted fitnesses are computed as follows:

$$F_1 = \left[1 + \left(\frac{F_1Raw}{n}\right)\right]^{-1} \tag{6}$$

$$F_2 = \left[1 + |C(G_t) - C(G_a)|\right]^{-1}, \tag{7}$$

$$F_3 = 1 - \frac{F_3}{n} \tag{8}$$

$$F_4 = 1 - F_4Raw \tag{9}$$

An adaptive weighting scheme was used that dynamically adjusts the weights during evolution in order to eliminate the need to manually search for useful objective weights. The rank weight for each objective $i \in O$ at generation t, denoted $w(t)_i$ was computed as:

$$w(t)_i = \frac{\sqrt{1.0 - \overline{F}(t-1)_i}}{\sum_{j \in O} \sqrt{1.0 - \overline{F}(t-1)_j}} \tag{10}$$

where O is the set of objectives, $\overline{F}(t-1)_i$ is the average adjusted fitness value for objective i at generation $t-1$. The value for $w(t)_i$ is used in place of w_i in (1).

When calculating the average geodesic path length, if there was no path between a vertex pair the length of the path was returned as the size of the vertex set, a value larger than any possible path. If the graph was empty, the worst possible fitness values were assigned. As a control to increase the quality of the models generated, evolved models may be executed more than once during evaluation, in this case the model is assigned the average fitness value per objective.

3.3 Construction of The Final Model

From the community and inter-community populations, a set of candidate community models, and a set of candidate

inter-community models are evolved. Exactly one model from each pool of candidates is needed to construct the final model, in order to select these models from their respective pools, each candidate from each pool was used to create 30 graphs that were compared to their respective target graphs. The comparisons were used to assign a final fitness value to each candidate community and inter-community model with respect to each objective, i. The models were then ranked according to these values. The rankings were then automatically assigned a weight, α_i, according to:

$$\alpha_i = \frac{\max(\mu_{ji}) - \min(\mu_{ji})}{\sum_{i=1} \max(\mu_{ji}) - \min(\mu_{ji})} \quad (11)$$

where μ_{ji} is the average fitness objective produced by the j^{th} evolved model. Each evolved model j was then assigned a final weighted rank sum, $RankSum_j$, which was used to automatically select a model from each pool to construct the final model (note that this value is used only for candidate model selection and not as a fitness value during evolution). $RankSum_j$ is computed as:

$$RankSum_j = \sum_i \alpha_i \times rank_{ji} \quad (12)$$

where $rank_{ji}$ is the rank of the j^{th} individual with respect to the i^{th} objective. Once the ranking process was completed the median, not the best, model was selected to be used for the final hierarchical model. Selection of the best model often led to selection of a model that was both over-fit, and not representative of the quality of the models the GP system was likely to produce. Thus, in selecting the median model we found our results improved.

3.4 Shape of the GP Tree and Language

Strong typing [21] was used to enforce a specific tree shape, which was evaluated in the following manner. The root node has three branches, each containing a list of actions. One branch for initialization, one branch which defines growth actions, and one branch which describes finalization actions. Operations in the *initialization* branch are responsible for adding vertices or specifying how vertices are to be added during the graph building process, these actions are executed once per evaluation. *Growth* operations are responsible for adding edges to the graph, and are executed n times per evaluation, where n is the desired number of vertices in the generated network. *Finalization* operations are any operations which require edges and vertices to be present in the graph, edge removal, for example. Figure 2 illustrates the tree structure.

The GP language includes a basic set of math operators, $\{+, -, *, \%\}$, Ephemeral random constants (ERC), boolean functions, $\{AND, OR, TRUE, FALSE\}$, IF structures, and the relational $<$ operator which takes two float arguments and returns a boolean (only the $<$ operator was included because swapping the arguments is equivalent to $>$). It also includes functions for adding and subtracting probabilities via the function $P(float)$ which accepts an argument of type *float*, and converting integer values to probabilities via $P(index)$ which accepts an argument of type *index*. These functions are defined at the end of this section. Initialization actions include a function to add all nodes to the network without any edges, one function to add all nodes and connect them in a ring, and one function to specify that one node is to be added per iteration. Finalization actions include a function to randomly rewire all edges in the network with an equal probability, a function to randomly rewire all edges, and a function to remove edges with a given probability. The terminal set includes *index* and *probability* types of fixed value, the value of the current node degree, the average degree of the active graph, the maximum degree of the active graph, the total vertex count, the final target vertex count, and the edge count of the active graph. There is one additional boolean function which returns true given a probability that can be used in any branch. The remaining growth actions are described below:

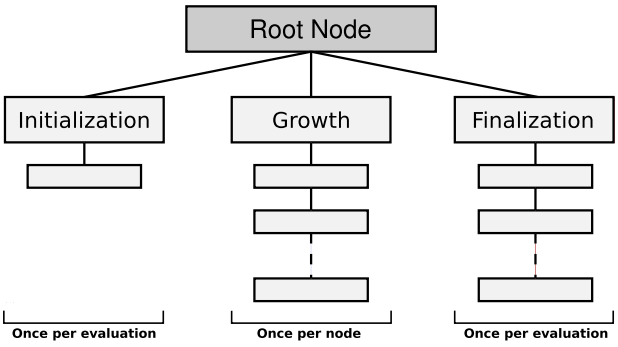


Figure 2: The shape of the trees used by the GP

- `GROW CREATE_TRIANGLE`: Creates a triangle that includes the current node in the active graph.
- `GROW CONNECT_W_PROB(prob)`: Connect to each node with probability *prob*.
- `GROW CONNECT_RAND`: Connects the current node in the active graph to some random node in the active graph.
- `GROW CONNECT_STUB(prob, bool)`: Connects an edge from the current node to a node that has previously executed `CONNECT_STUB`. Connects nodes to others which have called this function. Nodes are connected to other nodes probabilistically or randomly according to priority based on the degree of the node being connected to. Once a node has been connected to it is removed from the queue of available nodes. If no nodes were available to the calling node then it is entered into the request queue of nodes available for connection.
- `GROW CONNECT_STUB_PERSIST(Prob, bool)`: Connects an edge as `CONNECT_STUB` does, but it always adds its own request to the request queue and it does not remove any requests it satisfies from the request queue.
- `DUPLICATE(prob)`: Connects the current node to the neighbours of a randomly selected node. With probability p the current node is also connected to the selected node. This is based on models of protein interaction and this behaviour is known to generate an exponential degree distribution [34].

The function $P(index)$ takes an integer value as an argument and returns a floating point value from a pre-defined list at the location specified by *index MOD* L, where L is the length of the list. If the value of *index* is negative then

the absolute value is used. The function $P(float)$ takes a floating point value and returns a value between $[0, 1]$. If the argument's absolute value is already within that range, it simply returns the absolute value of the argument. If the absolute value of the argument is greater than one then it computes $P(|float|)$ and behaves in the same way as $P(index)$.

4. EXPERIMENTATION AND RESULTS

In order to ascertain how well the evolved model compares to the WS model and the BA model, the parameters of these models were tuned to fit the given data as well as possible via the methods described in the next subsection. Once the models were fit to the data, they were compared to the cortical network with respect to the average geodesic path length, transitivity, number of paths of all lengths between all node pairs. To do these comparisons 50 graphs were generated with the evolved, BA, and WS models respectively, because graph models are probabilistic in nature.

4.1 Parameter Tuning

The Barabási-Albert model is primarily concerned with matching the degree distribution of real-world networks [4], and so this the feature that we will focus on tuning. There are two parameters that can be adjusted, the power of the preferential attachment α_{BA} and the number of edges added per iteration, m. By adjusting the power we can influence the shape of the degree distribution, and adjusting m will translate it. The two parameters were tuned by a linear search of parameter combinations beginning with $m = 1$, and $\alpha_{BM} = 0.1$ and incrementing them by 1 and 0.1 respectively. Fifty graphs were generated with each parameter combination and the tuning process was terminated when the difference between the cumulative average degree distribution histogram from the fifty graphs and the cortex network was minimized. The difference was measured via the K-S test statistic and the final values chosen were $\alpha_{BA} = 0.3$ and $m = 11$. This combination produced a K-S test statistic of $D = 0.1$ and a *p-value* of 0.9899, indicating this was a very good fit.

In tuning the small-world model the value $p_{rew} = 0.08$ was taken from [36] in which the authors were fitting the same model to the same data, and the number of neighbours for the initial lattice was set at 10 to most closely match the number of edges in the cortex network.

4.2 Evolving A Cortical Model

A hierarchical model was generated as described in Section 3.1. The parameters used to evolve the model are listed in Table 1.

4.3 Performance of the Evolved Model

The final model constructs communities by initializing them with a ring of nodes, and then proceeds to create dense interconnections via the `DUPLICATE` function, as well creating some heavily connected hubs with the `CONNECT_STUB_PERSIST` function. It then probabilistically connects all pairs of nodes, and finally rewires a portion of the edges. The inter-community model, responsible for joining the communities, connects nodes by triangles and features hubs created again with a combination of the `DUPLICATE` and `CONNECT_STUB_PERSIST` functions, it also adds some edges probabilistically between all nodes. Finally, it probabilistically removes some edges. The combination of these models produces graphs with dense

Table 1: GP parameters.

Parameter	Value
Initialization Method	Koza's 'grow' method
Grow Min	3
Grow Max	5
Population Size	200
Generations	150
Selection	Tournament: k=3
Crossover	Subtree Crossover: 0.95
Mutation	Grow: 0.2, linearly decreasing, min depth = 1, max depth = 4
Runs	50

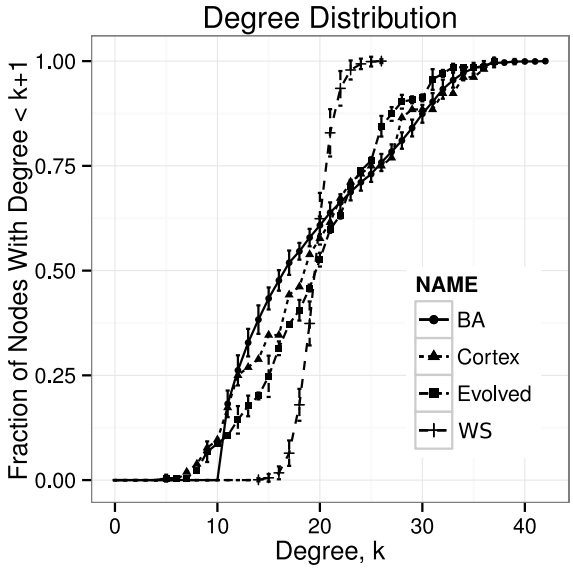


Figure 3: Cumulative degree distributions of the models compared to the cortical network

clusters, prominent hubs, and less dense inter-community connections which are joined with hubs. It also features short-cuts in the form of random edges between communities. The evolved model possess features of both WS model as well as the BA model. The average cumulative degree distributions of 50 graphs produced by each model are plotted against the cumulative degree distribution of the cat cortex in Fig. 3. The degree distribution of the WS graphs are clearly dissimilar to the cortical network, while the BA distribution and the evolved model distributions are very similar. A K-S test comparing the evolved model's average degree distribution to the cortical network's gives a test statistic of $D = 0.1053$, with *p-value* of 0.9844 which is similar to the fit of the BA model. Even more striking is how the degree distribution of a single evolved graph looks against the cortical degree distribution, as seen in Fig. 4. The cortical network distribution has a periodic look to it which is different than the unimodal distributions produced by the BA and WS models. The evolved model is capable of producing these unusual distributions which have reoccurring degree frequencies, similar to the distribution observed in the cortical model.

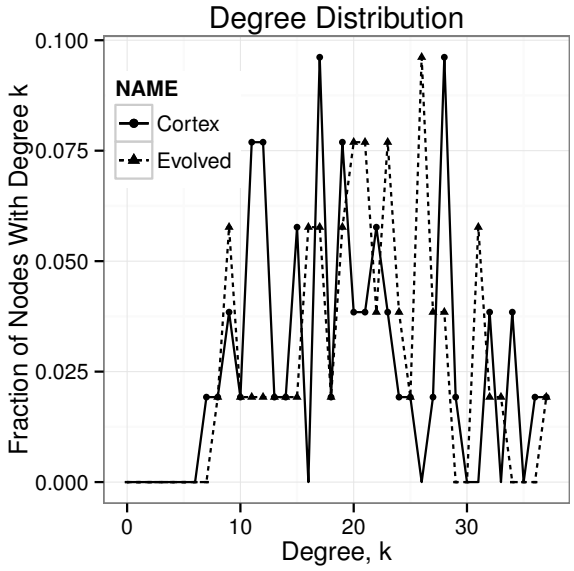


Figure 4: A single degree degree distribution produced by the evolved model vs. the cortical network

Table 2: Properties of graphs generated by the BA model compared to the cortical network, t.

	Min.	Q1	μ	Q2	Max.	$(\mu - t)^2$
$\|E\|$	506	506	**506**	506	506	$8.10{\cdot}10^{1}$
Trans.	0.47	0.48	**0.49**	0.49	0.50	$9.71{\cdot}10^{-3}$
Diam.	2	3	**2.92**	3	3	$6.40{\cdot}10^{-3}$
Avg. geo.	1.62	1.62	**1.62**	1.62	1.62	$2.48{\cdot}10^{-4}$
Comm.	2	3	**3.52**	4	5	$2.70{\cdot}10^{-1}$

Tables 2, 3, and 4 show how the models perform with respect to the size of the edge set $|E|$, transitivity (clustering coefficient), diameter, average geodesic path lengths, and the number of communities in the graphs. The far right column shows the squared error between the average values and the values found in the cortical network t. If the error value is bold it means it is at least as small as the smallest error of any of the models. The BA model produces the greatest errors, while the WS model is the closest in terms of edges and is able to consistently match the diameter of the cortical network. However, the evolved model also consistently matches the diameter, and is more similar to the cortical network than the other models with respect to these properties than the other models except the edge count where it falls between the BA and WS models. For reference, the cortical network has $|E_c| = 515$ edges, a transitivity of $C_c = 0.585$, a diameter of 3, an average geodesic path length of $l_c = 1.636$ and 3 communities.

It is thought that the number of alternate shortest paths between any two node pairs (i, j) may have an important impact on information processing within the cortex [36], this value is called the multiplicity, $M_{i,j}$. A good cortical model should be capable of generating similar frequencies of geodesic paths between node pairs. Fig. 5 shows how the average frequency of multiplicity values in 50 graphs generated by each model compare to the cortical network.

Table 3: Properties of graphs generated by the WS model compared to the cortical network, t.

	Min.	Q1	μ	Q2	Max.	$(\mu - t)^2$
$\|E\|$	520	520	**520**	520	520	$\mathbf{2.50}{\cdot}10^{1}$
Trans	0.47	0.52	**0.53**	0.54	0.56	$3.22{\cdot}10^{-3}$
Diam	3	3	**3**	3	3	**0**
Avg. geo.	1.61	1.61	**1.62**	1.62	1.62	$4.30{\cdot}10^{-4}$
Comm.	3	3	**3.72**	4	4	$5.18{\cdot}10^{-1}$

Table 4: Properties of graphs generated by the evolved model compared to the cortical network, t.

	Min.	Q1	μ	Q2	Max.	$(\mu - t)^2$
$\|E\|$	450	546	**523**	546	546	$6.40{\cdot}10^{1}$
Trans.	0.55	0.55	**0.55**	0.55	0.55	$\mathbf{1.04}{\cdot}10^{-3}$
Diam.	3	3	**3**	3	3	**0**
Avg. geo.	1.61	1.61	**1.64**	1.61	1.72	$\mathbf{1.81}{\cdot}10^{-5}$
Comm.	3	3	**3**	3	3	**0**

While all three models reasonably reproduce the frequencies of path lengths of one and two, only the evolved model produces a number of paths of length three comparable to those observed in the cortical network. Longer cortical pathways generally travel between different functional areas of the cortex [36], and in the evolved model graphs they travel between different communities. This is a feature the BA and WS models lack, and it is an important feature of the cortical networks given that these longer paths may be responsible for multisensory modulation and integration [36].

Another property which describes how elements of a network communicate is the betweenness centrality, it is affected by community structure and is an important property with respect to cortical networks [16, 32]. The mean betweenness was measured in the fifty graphs generated by each model, and the average of those scores was then taken. The BA and WS models both generate graphs with average betweenness scores more than twice as high as the cortical network, while the evolved model's average betweenness score is within one standard deviation of the cortical network. Table 5 lists the results.

Table 5: Betweenness centrality comparison

Model	μ	σ
Evolved	**16.31**	1.22
BA	37.85	0.144
WS	43.03	0.70
Cortex Mean Betweenness = **16.21**		

Lastly, Figures 6(a), 6(b), 6(c), and 6(d) show the cortical network, and examples of one randomly selected BA, WS and evolved model graph. The node sizes in the plots are proportional to their degree, and the nodes were positioned using the Fruchterman-Reingold algorithm [15]. It is possible to visually distinguish the clustered organization of the cortical network as well as the network generated by the evolved model and how they differ from the BA and WS graphs.

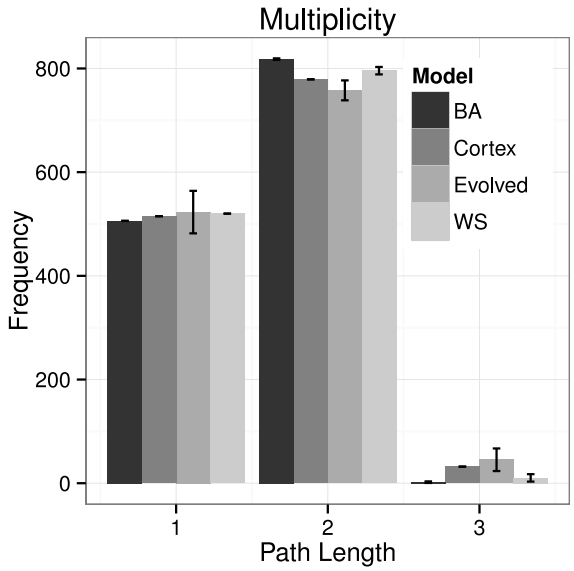
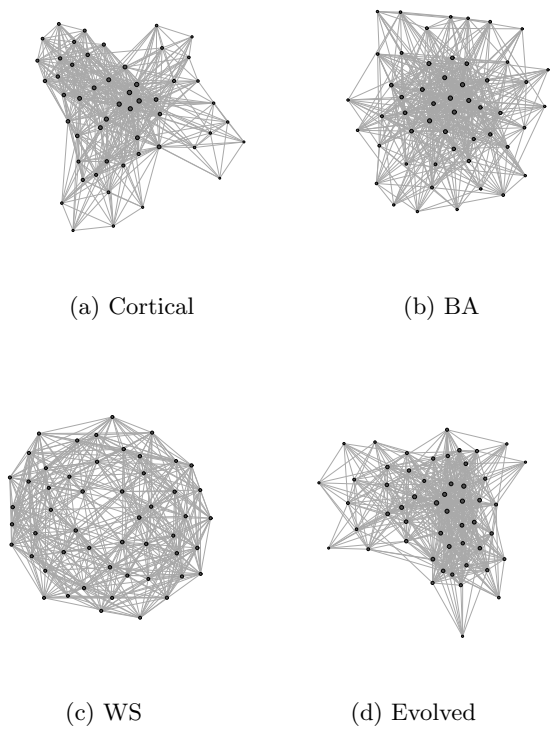


Figure 5: The number of shortest paths between all node pairs i, j in all models versus the cortical network.



(a) Cortical (b) BA

(c) WS (d) Evolved

Figure 6: The cortical network and a graph generated by each model.

5. CONCLUSION

The cortex is a complicated structure which possesses an inhomogeneous distribution of connections to other cortical areas, with communities of nodes more densely connected than others. This is reflected in the number and length of communication paths through the cortex. The organization of the communication pathways is important to healthy brain function, and classification and modelling of this behaviour has led to advancements in identifying unhealthy or injured brains. However, the important community structure of the cortex is not modelled by existing algorithms in common use and it is unclear how to properly design or select better algorithms. Recent work has shown that GP is a promising method for automatically generating graph models robust to any real-world data, such as cortical networks, especially in the case where good algorithms are unknown. This paper proposed the first GP system for the automatic inference of graph models capable of generating graphs which exhibit a strong community structure, and it was applied to infer a model of the cat cerebral cortex.

There are some challenges ahead to improving the process of inferring graph models which exhibit community structure, such as removing the dependence on an algorithm external to the GP. However, the proposed GP was able to evolve a model able to generated graphs more similar to the cortical network than existing models with respect to important features of the cortical network. Future work will focus on improving the process of inferring models of community structure, as well as exploring the breadth of applications which could benefit from the automatic inference of models.

6. REFERENCES

[1] R. Albert and A. Barabási. Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1):47–97, 2002.

[2] G. M. Ames, D. B. George, C. P. Hampson, A. R. Kanarek, C. D. McBee, D. R. Lockwood, J. D. Achter, and C. T. Webb. Using network properties to predict disease dynamics on human contact networks. *Proc. R. Soc. B*, 2011.

[3] A. Bailey, M. Ventresca, and B. Ombuki-Berman. Automatic generation of graph models for complex networks by genetic programming. GECCO '12, pages 711–718, New York, NY, USA, 2012. ACM.

[4] A. L. Barabási and R. Albert. Emergence of scaling in random networks. *Science (New York, N.Y.)*, 286(5439):509–512, Oct. 1999.

[5] D. S. Bassett, E. Bullmore, B. A. Verchinski, V. S. Mattay, D. R. Weinberger, and A. Meyer-Lindenberg. Hierarchical organization of human cortical networks in health and schizophrenia. *The J. of Neurosci.*, 28(37):9239–9248, 2008.

[6] P. Bentley and J. Wakefield. Finding acceptable solutions in the pareto-optimal range using multiobjective genetic algorithms. In *Soft Computing in Engineering Design and Manufacturing*, pages 231–240. Springer, June 1997.

[7] S. Bergen and B. J. Ross. Aesthetic 3d model evolution. In *Evolutionary and Biologically Inspired Music, Sound, Art and Design*, volume 7247 of *LNCS*, pages 11–22. Springer, 2012.

[8] E. Bullmore and O. Sporns. Complex brain networks: graph theoretical analysis of structural and functional

systems. *Nature Reviews Neuroscience*, 10(3):186–198, 2009.

[9] I. Chakravarti, R. Laha, and J. Roy. *Handbook of methods of applied statistics*, volume 1 of *Wiley series in probability and mathematical statistics*. Wiley, 1967.

[10] D. Corne and J. Knowles. Techniques for highly multiobjective optimisation: Some nondominated points are better than others. In *Proc. GECCO 2007*, pages 773–780. ACM Press, 2007.

[11] F. De Vico Fallani, R. Sinatra, L. Astolfi, D. Mattia, F. Cincotti, V. Latora, S. Salinari, M. Marciani, A. Colosimo, and F. Babiloni. Community structure of cortical networks in spinal cord injured patients. Proc. EMBS, pages 3995 –3998, 2008.

[12] S. F. Community detection in graphs. *Phys. Reports*, 486(3-5):75 – 174, 2010.

[13] R. Flack. The robgp genetic programming system, 2011. http://robgp.sourceforge.net/.

[14] R. W. J. Flack and B. J. Ross. Evolution of architectural floor plans. Proc. EvoApplications'11, pages 313–322. Springer-Verlag, 2011.

[15] T. M. J. Fruchterman and E. M. Reingold. Graph drawing by force-directed placement. *Softw. Pract. Exper.*, 21(11):1129–1164, Nov. 1991.

[16] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, 2002.

[17] B. J. He, J. M. Zempel, A. Z. Snyder, and M. E. Raichle. The Temporal Structures and Functional Significance of Scale-free Brain Activity. *Neuron*, 66:353–369, 2000.

[18] Y. He, A. Dagher, Z. Chen, A. Charil, A. Zijdenbos, K. Worsley, and A. Evans. Impaired small-world efficiency in structural cortical networks in multiple sclerosis associated with white matter lesion load. *Brain*, 132(12):3366–3379, Dec. 2009.

[19] S. Hill, W. Y., R. I., S. F., and M. H. Statistical connectivity provides a sufficient foundation for specific functional connectivity in neocortical neural microcircuits. Sept 2012.

[20] M. Kaiser, R. Martin, P. Andras, and M. P. Young. Simulation of robustness against lesions of cortical networks. *Euro. J. of Neurosci.*, 10:3185–3192, Aug 2007.

[21] D. J. Montana. Strongly typed genetic programming. *Evol. Comput.*, 3(2):199–230, June 1995.

[22] T. Nepusz, L. Négyessy, G. Tusnády, and F. Bazsó. Reconstructing cortical networks: Case of directed graphs with high level of reciprocity. In B. Bollobás, R. Kozma, and D. Miklós, editors, *Handbook of Large-Scale Random Networks*, volume 18 of *Bolyai Society Mathematical Studies*, pages 325–368. Springer, 2008.

[23] M. Newman. *Networks: An Introduction*. Oxford University Press, Inc., New York, NY, USA, 2010.

[24] M. E. J. Newman. The structure and function of complex networks. *SIAM Review*, 45:167–256, 2003.

[25] M. E. J. Newman. Modularity and community structure in networks. *PNAS*, 103(23):8577–8582, 2006.

[26] B. J. Prettejohn, M. J. Berryman, and M. D. McDonnell. Methods for generating complex networks with selected structural properties for simulations: A review and tutorial for neuroscientists. *Frontiers in Comp. Neurosci.*, 5:11, 2011.

[27] B. J. Ross. The evolution of higher-level biochemical reaction models. *Genetic Programming and Evolvable Machines*, 13(1):3–31, Mar. 2012.

[28] J. W. Scannell, C. Blakemore, and M. P. Young. Analysis of connectivity in the cat cerebral cortex. *J. of Neurosci.*, 15:1463, 1995.

[29] O. Sporns, C. J. Honey, and R. Kötter. Identification and classification of hubs in brain networks. *PLoS ONE*, 2(10):e1049, 10 2007.

[30] O. Sporns, G. Tononi, and R. Kötter. The human connectome: A structural description of the human brain. *PLoS Comput Biol*, 1(4):e42, 09 2005.

[31] O. Sporns and J. Zwi. The small world of the cerebral cortex. *Neuroinformatics*, 2:145–162, 2004.

[32] C. Stam and J. Reijneveld. Graph theoretical analysis of complex networks in the brain. *Nonlinear Biomedical Physics*, 1(1):3, 2007.

[33] C. J. Stam and E. A. de Bruin. Scale-free dynamics of global functional connectivity in the human brain. *Human Brain Mapping*, 22(2):97–109, 2004.

[34] A. Vazquez, A. Flammini, A. Maritan, and A. Vespignani. Modeling of protein interaction networks. *ComplexUs*, 1:38–44, 2003.

[35] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, 1998.

[36] G. Zamora-López, C. Zhou, and J. Kurths. Graph analysis of cortical networks reveals complex anatomical communication substrate. *Chaos*, 19(1):015117, 2009.